X-562-67-202

NASA TM X- 55843

A GENERAL-PURPOSE ON-BOARD PROCESSOR FOR SCIENTIFIC SPACECRAFT

Hartrsn I

oct 68

Hardware

X562-68-387

388

JULY 1967

GPO PRICE $ ____________

CFSTI PRICE(S) $ ____________

Hard copy (HC) 3.00

Microfiche (MF) .65

ff 653 July 65

GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

X-562-67-202

A GENERAL-PURPOSE ON-BOARD PROCESSOR FOR SCIENTIFIC SPACECRAFT

F. Styles
T. Taylor
M. Tharpe
C. Trevathan

Space Electronics Branch
Information Processing Division

July 1967

GODDARD SPACE FLIGHT CENTER
Greenbelt, Maryland

3210/2169/2120

A GENERAL-PURPOSE ON-BOARD PROCESSOR FOR SCIENTIFIC SPACECRAFT

F. Styles, T. Taylor, M. Tharpe, and C. Trevathan
Space Electronics Branch
Information Processing Division

ABSTRACT

This paper describes a general-purpose stored program digital computer being developed by members of the Space Electronics Branch for use on scientific spacecraft. The computer, called the on-board processor (OBP), has been designed to meet the objectives of multimission applicability, increased experiment capability, and simplified software construction with reliability being emphasized at both the system and component levels. This report includes a description of the processor's grammatically structured machine language together with a delineation of the central processing unit, input/output philosophy, and modular memory. Proposed circuitry and packaging techniques for both the engineering model and flight systems are presented.

CONTENTS

ILLUSTRATIONS

INTRODUCTION

Traditionally, spacecraft data-handling systems have performed a minimum of on-board data processing and have returned all experiment data for analysis by the experimenter. Space scientists have been reluctant to discard data before transmittal to the ground on the basis of criteria established before launch and have been unable for the most part to adjust their information-gathering apparatus to meet the situations encountered without compromising reliability with complex flight hardware. At present, many arguments exist for developing more extensive on-board data-processing techniques. The advent of microelectronics and the expense of ground-processing the enormous volume of data being returned from space have given considerable impetus to the idea of flying systems capable of reducing the data to be transmitted to "information" in the true sense of the word. Further, as various phenomena are investigated in more and more detail, increasingly more discriminating measurements must be transmitted within the bandwidth constraints imposed by spacecraft communications systems. The use of such devices as logarithmic counters by individual experiments has aided this data-transmitting effort to some extent, but the fact remains that the capacity of present spacecraft communications systems has been reached.

To circumvent this "information return impasse", the Space Electronics Branch is developing an on-board data-processing system in the form of a computer which will service individual experiment and spacecraft sensors to provide a more efficient data stream through data compression and the optimization of the data format. Since the computer can be reprogrammed and has a large computational capacity, any degree of on-board processing can be performed by reprogramming from the ground after launch. In this way a spacecraft can be launched with a simple data format and, as the characteristics of the phenomena and the behavior of the instruments in orbit are ascertained, additional degrees of data processing can be implemented.

This computer will be able to control the sampling and formatting of telemetry data, provide data storage, and perform lengthy and sophisticated computations for use in processing experiment, attitude control, and other spacecraft signals. The computer memory will also provide storage for commands to be acted upon after specified time intervals or upon certain inputs from the spacecraft subsystems and/or experiments. In addition, applying this on-board computer to prelaunch and postlaunch checkout of the spacecraft and experiments should greatly increase the effectiveness of those phases of spacecraft operations.

GENERAL DESCRIPTION

The on-board processor is being developed specifically for space application. The system is modular and consists, in a minimum configuration, of one central processor unit (CPU), one memory unit, and one input/output (I/O) unit, all connected to a common data bus. The interface circuits will be designed so that the data bus will be impervious to a significantly probable single component failure in an input or output circuit.

A CPU will be composed of about 1200 microcircuits and will dissipate approximately 3 watts of power. The unit is organized as an 18-bit parallel binary machine with a 7.5 μsec add time, one full-length index register, and hardware multiply/divide. Each memory module will have 8192 18-bit words. Power dissipation by the memory will be proportional to usage, with close to 5 watts for continuous access and about 0.75 watt required for standby operation. The addition of up to eight memory modules, giving a storage capacity of 65,536 18-bit words, requires very little additional power. The complexity and size of the I/O unit will depend entirely upon mission requirements because an I/O unit will be specifically tailored for each mission. However, if no analog to digital conversion is required, an I/O unit should not exceed the CPU in size or power requirements.

The memory module and CPU will each have approximate dimensions of 3 by 6 by 10 inches. Thus a minimum system of one CPU, one 8K memory unit, and one I/O unit would occupy about 540 cubic inches and would dissipate from 6 to 14 watts of power, depending upon memory usage. The capability and reliability of the system can be expanded by adding more memory and a standby CPU to be switched into service upon failure in the first CPU. It is significant that this more reliable system with increased capabilities (e.g., 65K memory and redundant CPU's) would still operate on about 15 watts of power and would be only 10 by 12 by 15 inches in size.

The Space Electronics Branch has the overall responsibility for the processor development effort. The engineering model of the CPU is being fabricated by Westinghouse Defense and Space Center, Aerospace Division under Goddard contract. General Precision, Inc., Librascope Group is responsible for the memory. The Space Electronics Branch is developing a general I/O. It is expected that these units will be operating as a system early in 1968.

OBJECTIVES

Design

In developing the spacecraft computer, several basic goals have been set. One major objective is to ensure that the computer will have multimission applicability. In other words, it will not be designed for a special mission but rather with the ability to adapt to large or small missions. This requirement will be met by defining the I/O interface between the CPU and the I/O box, and then tailoring the box for a specific mission. The CPU will have high-speed computational capability so that lengthy operations such as statistical analyses will be possible. Thus, through the use of such on-board data processing, a second major objective, that of increased experiment capability, can be met.

A third major design objective is the development of a software system which will, by its simplicity, allow efficient user program construction and debugging. To facilitate programming and to enable the program to be somewhat self-documenting, the basic machine language is grammatically structured restricted English. The instruction set is such that software simulators can be easily implemented on existing general-purpose stored-program computers. A comprehensive executive system will maintain control over subprograms, maintain interrupt priority, and automatically check, through a real-time clock interrupt, to ensure recovery from problem areas such as infinite loop execution.

Reliability

Ultra-reliable is a necessary description for a system of this sort which, for many applications, will be operating in an environment untenable by man. Reliability is being emphasized in the design of the processor at both the system and component levels.

At the system level, the modular organization enables reliability to be enhanced by the ease with which redundant functional subsystems may be interconnected. Figure 1 is a configuration of multiple memory, central processor, and I/O units. For such a system, any number of spare modules may be normally connected and powered on by ground command as failures occur in operating modules. For example, switching power from a malfunctioning memory to a spare, preloaded with the operational program, would result in 100 percent system repair.

Steps taken to assure maximum reliability at the component level are:

- Maximum use is made of monolithic integrated circuits.

- All circuit components will operate well below rated values to minimize failures due to electrical stresses.
- Only tried and proven reliable circuit techniques and components will be used.
- All active elements will be burned in, to detect early failures and to minimize drift due to aging.
- The number of electrical connections will be minimal with all internal connections either welded or soldered.
- All electronic subassemblies, including the memory planes, will be encapsulated.

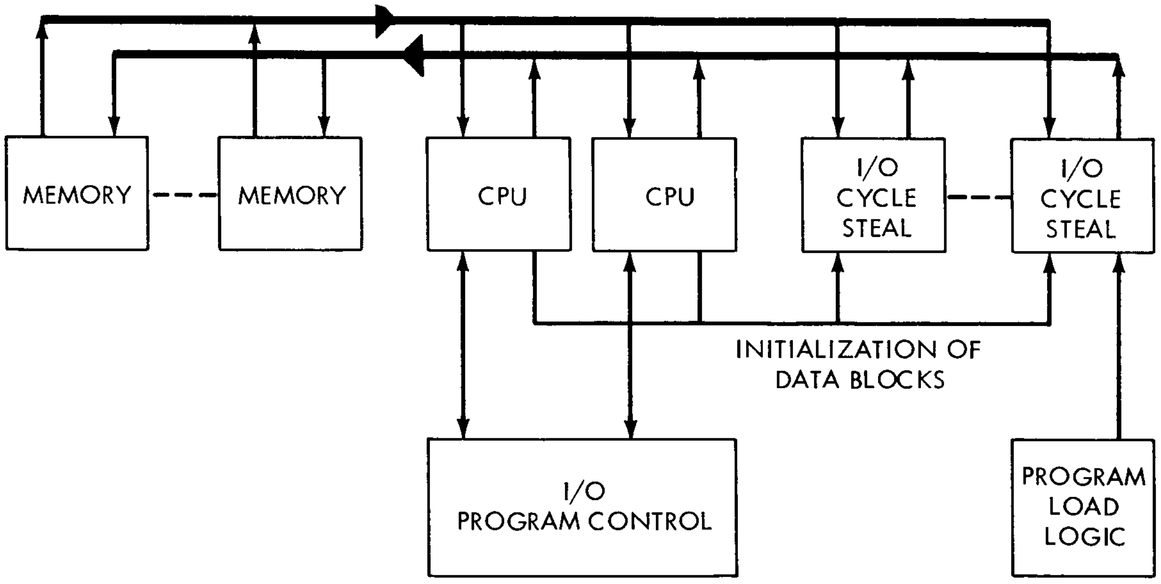


Figure 1. On-Board Processor, Functional Block Diagram

CENTRAL PROCESSOR UNIT

The central processor of the on-board computer employs a fully parallel adder and parallel data transfers between registers and at the CPU-I/O interface. Data words and instructions are 18 bits in length with negative numbers being represented in two's complement form. Addressable hardware registers include a 16-bit subscript register, an 18-bit accumulator, an 18-bit extension of the accumulator, a 4-bit memory page register, an 18-bit storage limit register which specifies read-only sections of memory, and a 6-bit scale register which represents the location of the binary point in fixed point data words. Figure 2 is a block diagram of the CPU.

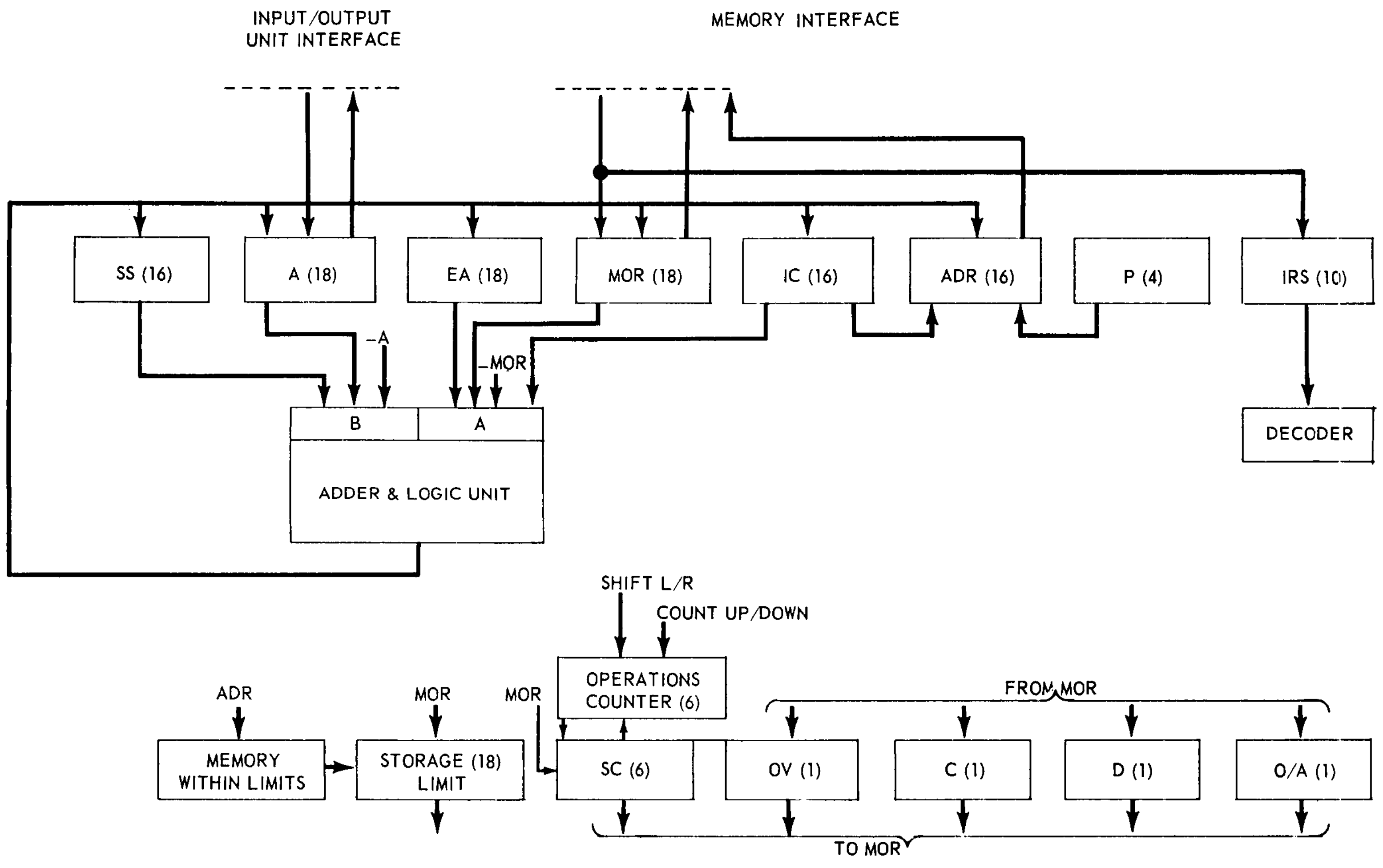


Figure 2. Central Processor Unit Block Diagram

An instruction has three fields and is formatted as follows:

$$\underbrace{B_{18}\ B_{17}\ B_{16}\ B_{15}\ B_{14}}_{\text{5-bit verb}}\ \Big|\ \underbrace{B_{13}}_{\text{1-bit subscript}}\ \Big|\ \underbrace{B_{12}\ B_{11}\ B_{10}\ B_{9}\ B_{8}\ B_{7}\ B_{6}\ B_{5}\ B_{4}\ B_{3}\ B_{2}\ B_{1}}_{\text{12-bit noun}}$$

At this stage of development, there are 47 instructions, 30 of which require an operand fetch. The other 17 instructions have a minor operation code in the address field of the instruction word. With 12 bits of address available, 4096 memory words are directly addressable. Memory size as large as 65,536 words requires a 4-bit page register which can be loaded and stored under program control and which is appended as four high-order bits to the 12-bit address field to form a full 16-bit address. If the subscript bit is set, a 16-bit subscript register is added to the address to form an effective address. All transfers are indirect, whereas all operand fetches are direct. The 18-bit storage limit register essentially reserves blocks in memory, each containing 128 words, that can be written into under program control. To implement this feature, the limit register is partitioned into two 9-bit fields. The first field specifies certain bits in the second field which must coincide with the nine high-order bits of the effective address for instructions which modify memory contents.

The CPU contains a 6-bit scale register which can be loaded and stored under program control. This register is used to maintain the binary point associated with numerical quantities at a program-controlled position for multiply and divide operations, thus freeing the programmer from this chore without sacrificing accuracy as would be the case with floating point operations. The processor also employs an addressable 18-bit extended accumulator, EA, which contains the remainder following a division (the quotient is contained in the accumulator) or the low-order 17 bits, plus sign, of a 34-bit product after a multiply operation. The 36-bit accumulator/EA is automatically shifted before a divide and after a multiply, with the shift length determined by the contents of the scale register. For most operations, the programmer will be concerned with the accumulator only; however, for those instances where direct control of the EA is desirable, e.g., double precision operations, instructions are available which allow extended accumulator loading from and storing into memory locations directly. For the most part, a programmer can view the machine as having only one 18-bit arithmetic register. Standard fractional arithmetic is possible by setting the scale register to zero and standard integer arithmetic is accomplished by making the scale register equal to 17.

Table 1 lists the functional registers.

Table 1
Central Processor Unit Functional Registers

Register	Symbol	Length (bits)
Memory operand	MOR	18
Memory address	MA	16
Instruction	IR	18
Instruction counter	IC	16
Subscript (index)	SS	16
Storage limit	SL	18
Page	P	4
Accumulator	A	18
Extended accumulator	EA	18
Scale	SC	6
Overflow	OV	1
Carry	C	1
Decision	D	1
Or/and	O/A	1

GRAMMATICAL MACHINE LANGUAGE

Machine Language Features

The grammatically structured machine language of the OBP was originally conceived by T. P. Gorman of the Telemetry Computation Branch, GSFC. Its main features are:

- Rules of programming are the rules of grammatical English.
- Vocabulary is restricted to the use of a limited number of phrases which either connect nouns or stand alone.

- Phrases correspond one for one with machine instructions.
- Nouns are the names of quantities or the names of operation sequences (e.g., subroutines) used in the course of a program.
- Numbers, octal or decimal, may be used directly to represent numeric quantities.
- All transfers are indirect for simple program relocation.
- User may insert punctuation as desired to make text readable.
- Programs are written in free (paragraph) form; additional comments may be entered parenthetically within the text, thereby making the program self-documenting.
- A memory protect feature prevents modification of instructions. Only nouns are modifiable. Subroutine return addresses are stored as though they were nouns. Therefore, once a program is loaded into memory, only data quantities will change and debugging is considerably simplified.

Instruction Set Description

In the following description, parentheses indicate the contents of the memory location or register, which they surround. For example, (M) reads "the contents of memory location M." See Table 1 (page 7) for register definition.

ACCUMULATOR

1. let M if M	Load A with (M).
2. let location M	Load A with M.
3. yield M	Store (A) in M.
4. set EA with M	Load EA with (M).
5. save EA in M	Store (EA) in M.
6. close EA with D	Shift (A) and (EA) left one bit and put (D) in the least significant bit of EA. Shifting is around the sign bit of EA and through the sign bit of A with OV being set if the sign of A is changed in the process.

SUBSCRIPT REGISTER	
7. use subscript M	Load SS with (M).
8. save subscript in M	Store (SS) in M.
9. step subscript by M	Add (M) to (SS).
10. if subscript not greater than M	If (SS) $\leq$ (M), set D = 1. If not, leave D unchanged.
ARITHMETIC	
11. plus M	Set (C) to zero. Add (M) to (A).
12. plus carry	If the previous arithmetic operation resulted in a carry from the most significant bit of A, setting (C) to 1, set (C) to zero and add 1 to (A).
13. minus M	Set (C) to zero. Subtract (M) from (A).
14. times M	Multiply (A) by (M) and position the 34-bit product plus sign in A, EA so as to maintain the binary point as indicated by (SC). In positioning, all shifting is around the sign bit of EA.
15. over M	Divide (A), (EA) by (M) and put the quotient in A and the remainder in EA, maintaining the binary point of both as indicated by (SC). As the dividend (A), (EA) are treated as a single 34-bit quantity plus sign with the sign bit of EA ignored.
LOGICAL	
16. anded with M	Logically "AND" (M) with (A).
17. ored with M	Logically "OR" (M) with (A).
18. eored with M	Logically "EOR" (M) with (A).
19. complemented	Replace (A) with its 1's complement.
20. negated	Set (C) to zero. Replace (A) with its 2's complement.
BIT MANIPULATION	
21. shifted by M	Shift (A) by (M)-bits, left for positive (M), and right for negative. Bits shift left through the sign bit of A, but when

21. shifted by M (continued)	a bit different in value from the original sign shifts into the sign position, OV is set. On right shift, the sign bit is copied into the vacated bit positions.
22. double shifted by M	Shift (A) and (EA) by (M)-bits, left for positive (M), and right for negative. Bits shift left through the sign bit of A and around the sign bit of EA. If a bit different in value from the original sign bit of A shifts into the sign position of A, OV is set. On right shift, the sign bit of A is copied into the vacated bit positions. Again, bits shift around the sign bit of EA.
23. cycled by M	Cycle (A) by (M)-bits, left for positive (M), and right for negative. A cycles onto itself and no bits are lost.
24. double cycled by M	Cycle (A) and (EA) by (M)-bits, left for positive (M) and right for negative. (A), (EA) are connected as a single 36-bit register which cycles onto itself and no bits are lost.
25. normalized	Clear SC. If (A), (EA) = 0, do nothing. If (A), EA $\neq 0$, shift (A), (EA) left until bit 18 $\neq$ bit 17 of A, and put the number of positions shifted into SC. Shifting is around the sign bit of EA.

TEST

26. is less than M	If (A) < (M), set (D) = 1. If not, leave (D) unchanged.
27. is equal to M	If (A) = (M), set (D) = 1. If not, leave (D) unchanged.
28. is greater than M	If (A) > (M), set (D) = 1. If not, leave (D) unchanged.
29. is positive if positive	If sign bit of A = 0, set (D) = 1. If not, leave (D) unchanged.

30. if overflow	If (OV) = 1, set (D) = 1. If not, leave (D) unchanged.
31. if parity odd	If parity of (A) is odd, set (D) = 1. If not, leave (D) unchanged.
32. is false	Complement (D).
33. and	"AND" present state of D with next test result.
34. or	"OR" present state of D with next test result.

TRANSFER

35. then go to M	If (D) = 1, branch to (M) and set (D) = 0. If (D) = 0, continue.
36. go to M return to M	Branch to (M).
37. transformed by M	This is the subroutine call. Store (IC) plus 1 in M; go to (M + 1).

CONTROL

38. pass	No operation.
39. halt	Halt.
40. execute M	Execute (M) as an instruction. Add 1 to (IC).

MISCELLANEOUS REGISTERS

41. set scale with M	Load SC with the six least significant bits of M.
42. let scale	Load the six least significant bits of A with (SC). The other 12 bits of A are set to zero.
43. set page	Load P withfour least significant bits of A.
44. reset OV	Make (OV) = 0.

INTERRUPT

45. exit	This instruction causes a program-controlled interrupt. The page, instruction counter, scale, storage limit, interrupt priority status, C, OV, D, and OA registers contents are saved in four consecutive memory locations. The next four locations set the above registers to the status desired.
46. resume from M	This instruction terminates each interrupt routine. The previous register status is restored from the first four locations of the interrupt under execution.

INPUT/OUTPUT

47. output to M*	(A) are output as data to the I/O unit.
let input from M*	The I/O unit loads one data word into A.
if I/O status of M*	The I/O unit loads one status word into A and sets up for a test.
connect to M*	(A) are output as a memory address to the I/O unit.

ASSEMBLER DIRECTIVES

1. subscripted	Add (SS) to the address of the noun which follows to form the effective address for the present operation.
2. it	This is a dummy operand used with LET and IF to initiate an operation without altering the accumulator.
3. be	This is used alone or with certain verbal phrases to close a sentence (i.e., end an operation sequence) leaving the result in A.
4. allocate N locations for M	Reserve N locations of storage for array M.

*Memory location M contains a function code which defines the I/O channel activity.

5. defining M	Assign an address to M indicating a position in the instruction sequence.
6. assign M to location N	Assign the fixed octal location N as the address of M.
7. preset M to N	Preset the constant M to the value N, where N may be octal or decimal.

Sample Programs

The following program computes the sum of the squares of 100 numbers:

```
START THE FOLLOWING PROGRAM AT LOCATION 1000. ALLO-
CATE 100 LOCATIONS FOR NUMBERS. (INITIALIZATION) LET 0
YIELD SUM. USE SUBSCRIPT 0.
DEFINING LOOP START, LET SUBSCRIPTED NUMBERS TIMES
SUBSCRIPTED NUMBERS PLUS SUM YIELD SUM. STEP SUB-
SCRIPT BY 1. IF SUBSCRIPT IS NOT GREATER THAN 99, GO TO
LOOP START.
```

It is assumed above that the array, NUMBERS, has data inserted before this program is executed.

The following program computes the sine of an angle (in radians) by summing terms of the Taylor series until 10 terms have been used or the next term is less in absolute value than a fixed number, EPSILON, whichever comes first.

```
DEFINING SINE, LET 1 YIELD N. LET 0 YIELD SINX. LET X
YIELD TERM. DEFINING LOOP, LET SINX PLUS TERM YIELD
SINX. LET N PLUS 1 YIELD PARTNO. LET IT PLUS 1 YIELD N.
LET IT TIMES PARTNO YIELD NUMBER. LET TERM TIMES X
TIMES X DIVIDED BY NUMBER NEGATED YIELD TERM. IF
TERM IS GREATER THAN EPSILON OR IF TERM NEGATED IS
GREATER THAN EPSILON AND IF N IS LESS THAN 10, THEN GO
TO LOOP.
HALT.
```

Among other things, this illustrates the comparative ease with which a loop may be terminated conditionally, depending on the value of a complicated logical statement.

INPUT/OUTPUT UNIT

Design objectives require that the on-board processor be applicable to a variety of scientific spacecraft programs. To be successful in this approach, a general and versatile technique for data transfer and interrupt processing must be developed. The intention is to tailor the computer's I/O scheme to accommodate a particular mission's individual and perhaps unique requirements, with little or no impact on the CPU and memory unit design.

With this objective in mind, the system design has been established to provide:

- Data transfer external to program execution (called the cycle steal mode)

- Data transfer completely under program control

In the first method, the I/O time shares the memory with the CPU, allowing continuous data input without interrupting CPU processing. The second, program-controlled data transfer, may be requested in two ways. The faster is to interrupt the program. Whenever an interrupt occurs, the processor (upon completion of the current instruction) leaves the program it is processing, goes to the routine associated with that particular interrupt, and there executes the appropriate input instructions. After the routine has been completed the processor returns to the original program. A slower type of program-controlled data transfer is one in which the processor periodically scans a register to determine if an input/output request has occurred. If so, the processor acts on the particular request according to its priority. This type of data transfer is feasible when instruction execution rates are fast compared to the input data rates.

Data Transfer Operation

Execution of the input/output instruction (instruction 47) will activate either a data output, a data input, a status word entry, or the initiation of a channel in the cycle steal mode. The I/O control system makes use of the contents of address M and the CPU accumulator to control which channel activity will take place. If a data transfer is to occur, (M) are interpreted by the I/O control unit and, as appropriate, a single 18-bit data word is transferred by way of the CPU accumulator. For initiation of a cycle steal mode, (M) will again specify the action and in this case the accumulator contains the starting address for the data block. In general, the control word addressed by M specifies channel, block length, function, etc.; however, the exact format will probably vary with each application. It should be noted that for program control of data transfers, the accumulator must be loaded prior to an OUTPUT TO M; likewise, the program must store the accumulator following a LET INPUT FROM M. Timing of I/O instruction execution assumes that the peripheral device is awaiting the data transfer so it is not necessary to implement an automatic timeout. This approach is acceptable since execution of the instruction is a result of an interrupt or other program-scanned data request.

As stated, data can be transferred external to program execution by a technique of stealing memory cycles. In this case the memory address for each block of data transfers is specified by the I/O control unit. This feature is implemented with one counting register to contain a 16-bit address and a second counting register containing the block length. These registers may be initialized by execution of the CONNECT TO M instruction or from an external source for original program loading. For each transfer the block length register is decremented by 1, and an interrupt is generated when the register count reaches zero.

Interrupt and Data Request Operation

A peripheral device may request processor attention by interrupting program execution or by setting a data transfer request line which is scanned periodically by the program. If the interrupt approach is used, the appropriate I/O instruction is executed as part of the interrupt subroutine. The system will be capable of defining up to 64 unique interrupts. For the data-request technique, the executive program will periodically input a special 18-bit data word which represents previous requests from a set of peripheral devices. Following the logical interpretation of this word, an appropriate I/O instruction will be executed.

Interrupt Description—An interrupt may be defined as the action which occurs when the execution of one program is pre-empted in favor of the execution of a second routine. For graceful recovery from an interrupt, the contents of all active unaddressable registers must be stored before entering the interrupt routine. This storage provides the necessary information for register restoration when the interrupted program is re-entered. Addressable registers are stored and entered as necessary in the interrupt subroutine.

For better understanding, the sequences which occur when entering into and exiting from an interrupt subroutine are described. Consider that interrupt N ($01_8 \leq N \leq 77_8$) has been allowed by the I/O unit to signal the processor control. Execution of the current instruction is completed, the instruction counter (IC) is stored in $N0_8$, and by special CPU control the unaddressed registers are stored in memory locations $N1_8$, $N2_8$, and $N3_8$ as shown in Table 2.

Table 2
Fixed Memory Location For Interrupt N*

Memory Address, NX	Address Contents
N 0	(Instruction Counter) at time of interrupt
N 1	(Storage Limit) at time of interrupt
N 2	(Scale, Page, O/A, D) at time of interrupt
N 3	(Interrupt Lockout Status) at time of interrupt
N 4	New Interrupt Lockout Status
N 5	New Scale, Page, O/A, D
N 6	New Storage Limit
N 7	GO TO M

*N represents the two most significant digits of a three-digit octal number; therefore, $01X \leq NX \leq 77X$

These same registers are then loaded with new values from fixed locations $N4_8$, $N5_8$, and $N6_8$. The interrupt subroutine is then entered by execution of a GO TO M instruction located in $N7_8$. Exit from the interrupt is accomplished by a RESUME instruction, which must be the final instruction in the subroutine. Execution of this command causes the contents of N 0, 1, 2, and 3 to be stored in the appropriate registers, and re-entry into the original program is accomplished as the IC register addresses the next sequential instruction.

Interrupt Priority—It may be desirable for several reasons to allow a second interrupt to have precedence over the processing of the first. In other cases the converse may be true; consequently, a level of priority must be established. This priority can be implemented by selectively locking out the interrupts which are not to be allowed during the execution of any given routine. Lockout information is dependent upon the program, with the appropriate status word transmitted to the I/O control unit as part of the interrupt entry and exit sequences previously described. With this data, the I/O unit can maintain control by passing the allowable interrupts and temporarily storing all others until higher priority processing is complete. It should be noted that priority levels could be hard-wired as a part of the I/O control logic; however, the proposed technique provides a means of re-ordering these levels of priority while the spacecraft is in orbit. Conceivably, interrupts from a malfunctioning device could be permanently inhibited by minor reprogramming.

FLIGHT MEMORY UNIT

The memory unit is a nondestructive readout woven plated-wire system, being developed by General Precision, Inc., Librascope Group,under Goddard contract NAS5-10077. Each memory unit has a data capacity of 8192 18-bit words. Total power required is 8.2 watts at a 400 x 10^3 words/sec write rate with power consumption decreasing linearly with duty cycle to less than 0.75 watt at standby. It occupies 170 cubic inches in dimensions of 9-1/2 by 6 by 3 inches. Properties of this memory system which make it ideally suited to requirements of the on-board processor are:

- A natural nondestructive readout mode
- Low drive currents (65-ma digit current, 200-ma word current)
- High speed, allowing a duty cycle of less than 0.25 at a 2-μsec write rate
- Equal word-read and word-write currents, providing a radical reduction in selection system complexity
- High output signal and signal-to-noise ratio (at least 10 mv and at least 3:1, respectively)
- Very simple temperature compensation through the -20 to +80°C operating range
- An intrinsically shock- and vibration-resistant woven structure

Memory Organization

The spacecraft memory has a stack aspect ratio of 1024 words by 144 bits. There are 1024 word lines and four sets of 36 plated-wire digit lines. When a word line is pulsed, word current flows through four sets of memory cells corresponding to four data words. One set of 36 digit drivers and sense amplifiers is commutated across the four sets of digit lines by decoding two address bits. The 36 digits are further divided into two groups of 18 bits each. Two additional address bits are decoded for the selection of one 18-bit half, the second 18-bit half, or both. Selection of one from among the 1024 word coils is accomplished by a 32 x 32 diode decoding matrix. The remaining 10 address bits are decoded to control the diode matrix.

During a read cycle, word current is pulsed through the selected word coil. The resulting sense voltage output developed on digit lines is coupled into the sense amplifier by the selected group of 18 or 36 commutation switches. The sense signal is amplified, discriminated, and then transmitted to the memory data register M.

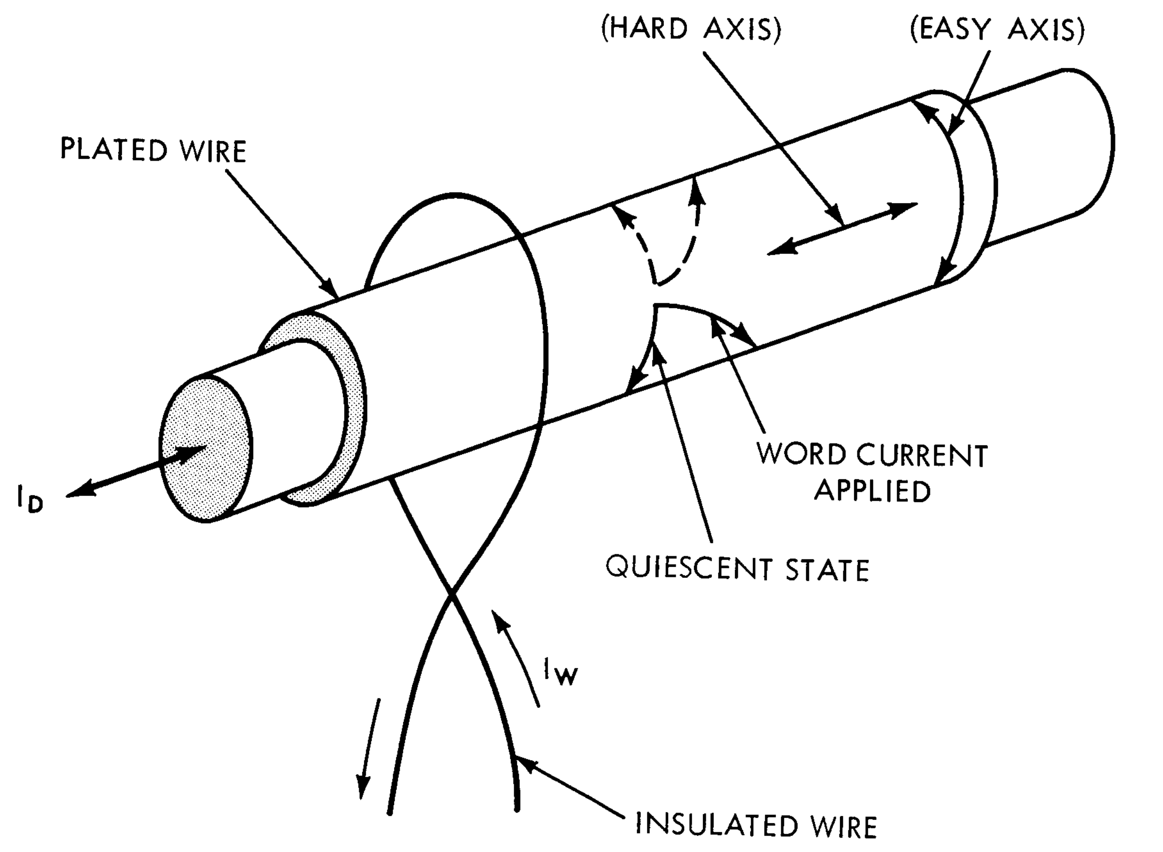
During a write cycle, digit current is switched on. It is positive or negative, depending upon "1" or "0" data input from M register. Digit current flows through the selected group of 18 or 36 commutation switches into the digit lines. While digit current is on, word current is pulsed through the selected word coils. At those junctions in the memory matrix where digit and word currents coincide, information is written.

In considering the electronic organization of the memory, emphasis has been placed on the use of integrated circuits. Logic gates, logic flip-flops, address decoder circuits, digit drivers, sense amplifiers, sense discriminators, and choppers are monolithic integrated circuits. Circuits such as word drive switches, current sources, and power drivers are implemented with discrete or hybrid circuits, since presently available monolithic circuits cannot meet current or voltage requirements. These discrete component circuits are all transformer-coupled and are normally biased off by the transformer short across the base-emitter junction of the transistor, so that none contributes to steady-state power dissipation.

Theory of Plated-Wire Operation

In writing, the word-drive current (I_W) in the insulated wires rotates the magnetization in the plated magnetic film from the easy toward the hard direction as shown in Figure 3. Digit-drive current (I_D) in the plated wire then tilts the magnetization vector away from the hard direction. When the word current is removed, the magnetization vector rotates to the easy axis in the direction determined by the polarity of the digit-drive current. The two binary storage states consist of the two possible directions of the easy-axis magnetization, with selection made by the tilting effect of the digit current. The digit current is removed last.

The memory is interrogated by pulsing the word lines. This signal rotates the magnetization in the plated wire toward the hard direction, inducing a voltage in the plated wire which now serves as the sense line. The polarity of this



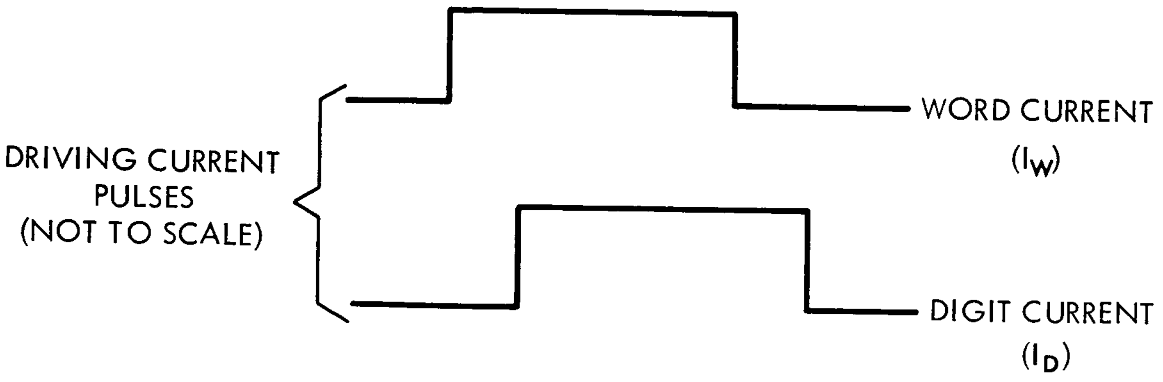


Figure 3. Memory Unit, Write-in Process

sense voltage is dependent upon the direction of the original easy-axis magnetization.

In nondestructive readout operation, the magnetization vector rotation is limited so that it falls back to its original orientation upon conclusion of the interrogate read pulse. Data are retained through a read cycle.

CIRCUITRY

Integrated circuits will be used extensively in the on-board processor. Circuits selected for the engineering model are the Fairchild 9040 series, a low-power diode transistor logic (LPDTL) employing a +5-volt power supply. The logic is packaged in 14 lead ceramic flat packs, and will operate at speeds up to 2.5 MHz with power dissipations of 1 mw per gate (50 percent duty cycle) and 4 mw per flip-flop. These circuit elements are readily available and have one of the lowest power-speed products of any logic type presently available in large quantities from stock. The 9040 series, a standard line which has been in existence for approximately 2 years, is well established and has a good reliability history.

PACKAGING

Engineering Model

An engineering model of the CPU and I/O units will contain small pluggable cards (1.2 by 4.4 inches) on which as many as 12 flat packs can be mounted. The flat packs are stacked two high, the top one rotated 90 degrees with respect to the one below it. Both packages are mounted 45 degrees with respect to true rectilinear for lead-access purposes. The printed circuit card contains a blade-type connector which mates with a split-type (tuning fork) female contact. The back end of the female connector consists of a wire-wrap post. The printed circuit cards are general purpose and all interconnections and intraconnections are performed on the backboard with an appropriate wire-wrap tool. The wiring lists will be prepared by computer and the wiring can be performed either under computer control or by hand. See Figure 4.

Approximately 120 of these cards will be required to fabricate the CPU. The proposed configuration will be four rows of 30 cards each, with the cards on 0.3-inch centers. The connector panel will have slotted card guide assemblies mounted into it and will be assembled into a housing with input/output connectors positioned at one end.

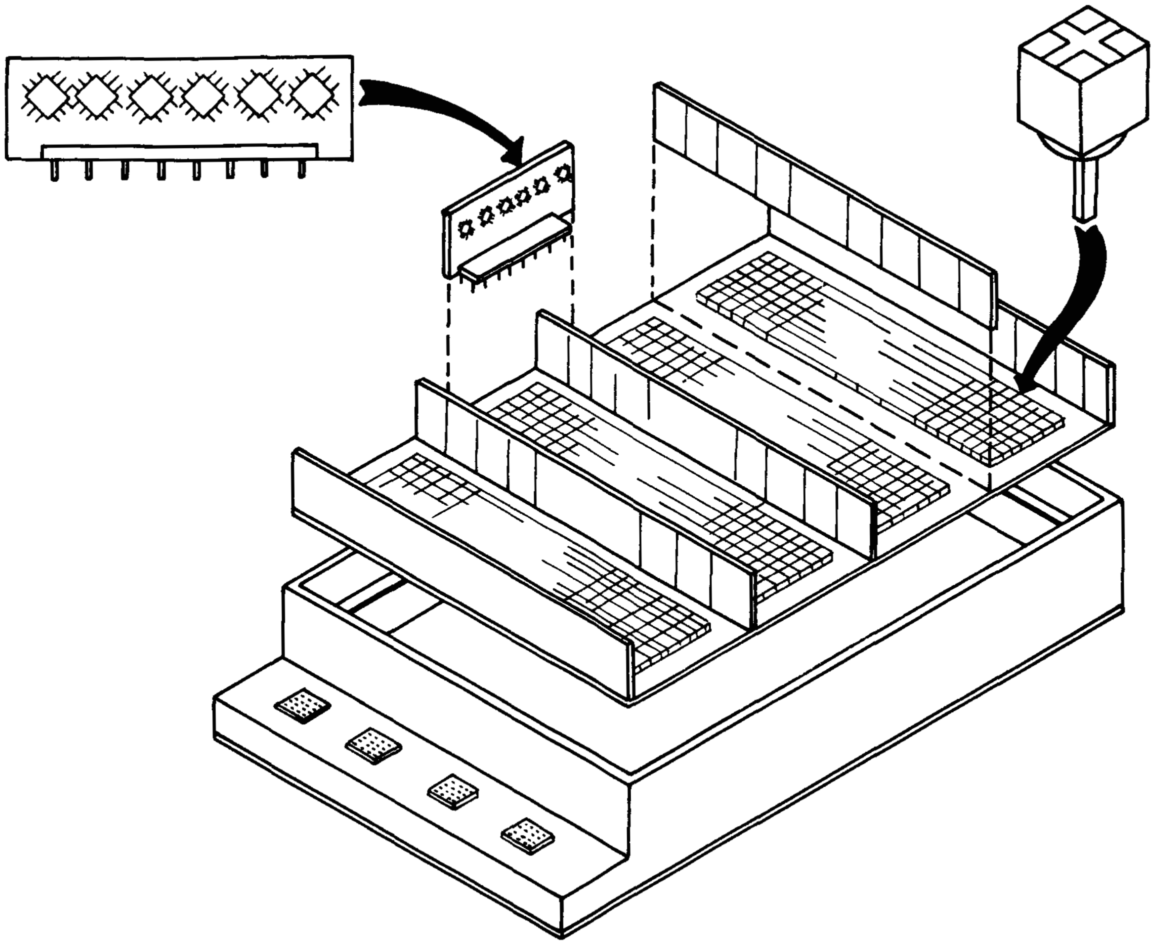


Figure 4. Engineering Model Packaging of CPU and I/O Units

A discussion of the memory packaging will not be given here due to the similarity of the packaging concepts for the engineering model and flight units.

Flight Model

The final packaging configuration, excluding the memory, will use stacked array modules containing up to 12 flat packs each, as shown in Figure 5. In a stacked array module, the flat packs are stacked on edge in a plastic comb-like header so that their leads project outward horizontally. Feed-through leads are provided in the comb spacers. Interconnections are made by nickel ribbon, welded to the adjacent flat-pack leads or spacer feed-throughs. The base of the header contains provisions for output leads.

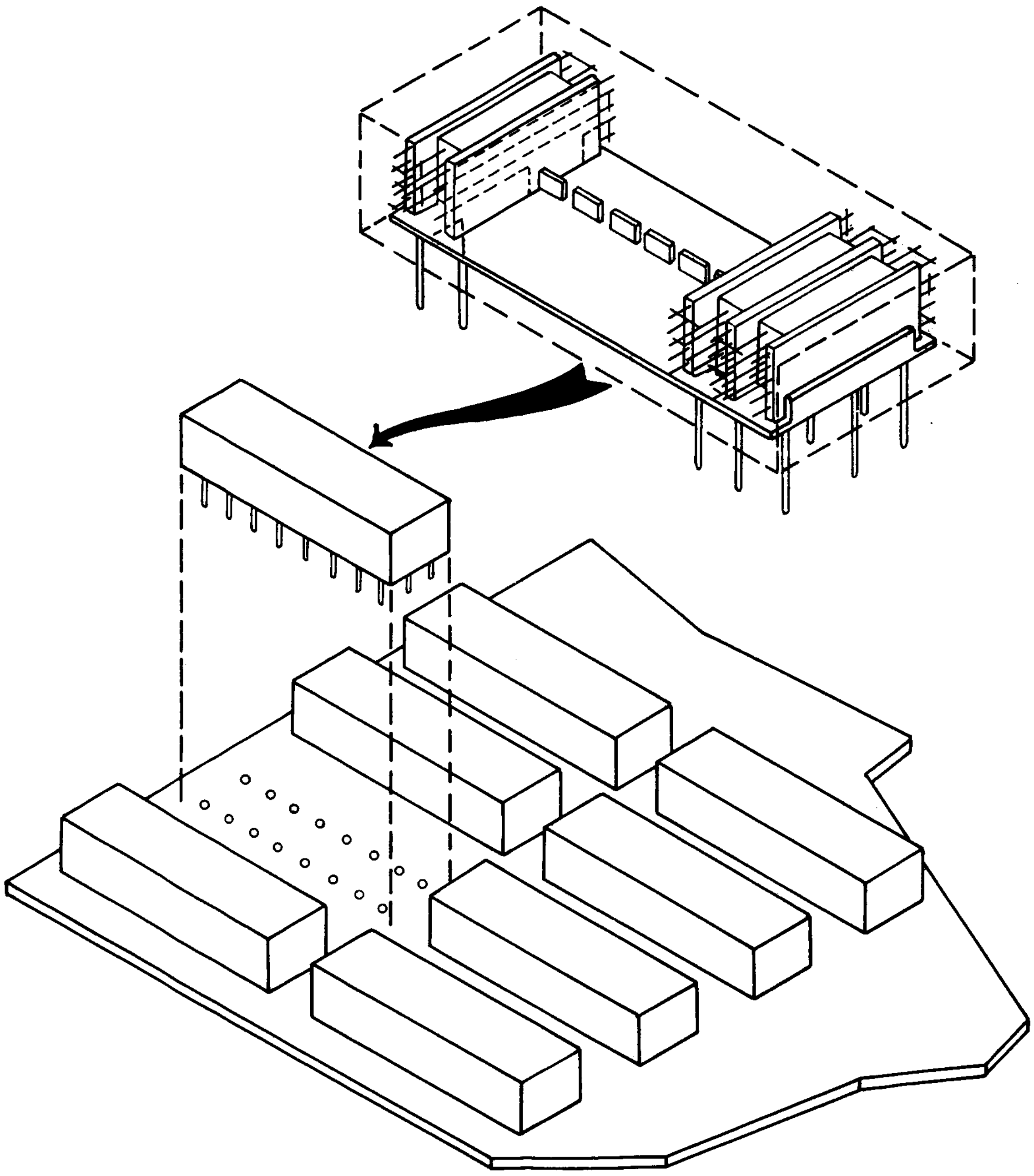


Figure 5. Flight Packaging of CPU and I/O Units

After the modules are assembled and tested, they are encapsulated in foam or epoxy. The potted module is 1-1/2 by 1/2 by 1/2 inches in size. These modules may be interconnected by various techniques such as double-sided plated-through-hole printed circuit boards, multilayer boards, or welded backboards. The overall CPU package volume using one of the above techniques will be approximately 144 cubic inches. The size of the I/O module is dependent upon the complexity of the mission, but generally it would occupy a volume equal to or less than the CPU.

In regard to the memory unit, fabrication of engineering and qualification models will precede production of flight units. Flight units will have physical features identical to those of the qualification model and with present design plans, each 8K,18-bit (or 4K, 36-bit) memory package has dimensions of 7.7 by 5.7 by 3 inches, and a total weight of 6.2 pounds. Not included in this package is the required power converter which would add approximately 20 percent to both weight and volume. However, a single power converter could provide power to several memory units and consequently total packaging can vary with overall system configuration.

Figure 6 shows that the memory stack is composed of eight memory planes; each plane contains two mats of plated digit wire, woven with insulated word-select wires. The total stack is sandwiched between the word-select electronics assembly and a pair of digit-select assemblies. Each of these assemblies are constructed with functional circuit modules which are individually encapsulated and potted. With different module types containing different circuits, maximum packaging density causes a variation in module dimensions. Module height is maintained constant to yield an optimum set of dimensions for the total assembly.

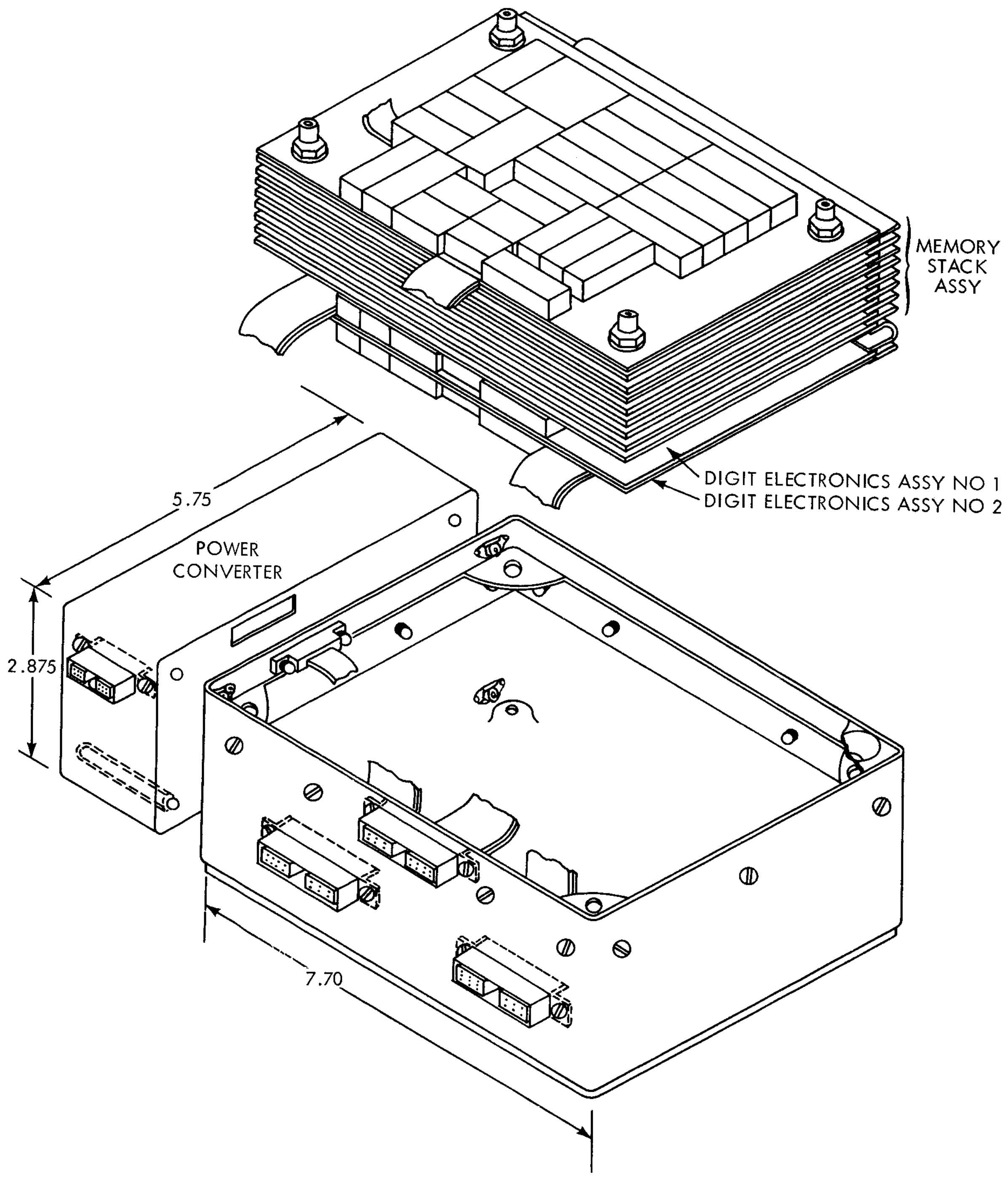


Figure 6. Memory Packaging